ABSTRACT OF THE DISCLOSURE

The invention provides an improved method and apparatus for creating a snapshot of a file system. In a first aspect of the invention, a "copy-on-write" mechanism is used. An effective snapshot mechanism must be efficient both in its use of storage space and in the time needed to create it because file systems are often large. The snapshot uses the same blocks as the active file system until the active file system is modified. Whenever a modification occurs, the modified data is copied to a new block and the old data is saved (henceforth called "copy-on-write"). In this way, the snapshot only uses space where it differs from the active file system, and the amount of work required to create the snapshot is small. In a second aspect of the invention, a record of which blocks are being used by the snapshot is included in the snapshot itself, allowing effectively instantaneous snapshot creation and deletion. In a third aspect of the invention, the state of the active file system is described by a set of metafiles; in particular, a bitmap (henceforth the "active map") describes which blocks are free and which are in use. The inode file describes which blocks are used by each file, including the metafiles. The inode file itself is described by a special root inode, also known as the "fsinfo block". The system begins creating a new snapshot by making a copy of the root inode. This copy of the root inode becomes the root of the snapshot. The root inode captures all required states for creating the snapshot such as the location of all files and directories in the file system, it. During subsequent updates of the active file system, the system consults the bitmap included in the snapshot (the "snapmap") to determine whether a block is free for reuse or

belongs to the snapshot. This mechanism allows the active file system to keep track of which blocks each snapshot uses without recording any additional bookkeeping information in the file system. In a fourth aspect of the invention, a snapshot can also be deleted instantaneously simply by discarding its root inode. Further bookkeeping is not required, because the snapshot includes it's own description. In a fifth aspect of the invention, the performance overhead associated with the search for free blocks is reduced by the inclusion of a summary file. The summary file identifies blocks that are used by at least one snapshot; it is the logical OR of all the snapmap files. The write allocation code decides whether a block is free by examining the active map and the summary file. The active map indicates whether the block is currently in use in the active file system. The summary file indicates whether the block is used by any snapshot. In a sixth aspect of the invention, the summary file is updated in the background after the creation or deletion of a snapshot. This occurs concurrently with other file system operations. Two bits are stored in the file system "fsinfo block" for each snapshot. These two bits indicate whether the summary file needs to be updated using the snapshot's snapmap information as a consequence of its creation or deletion. When a block is freed in the active file system, the corresponding block of the summary file is updated with the snapmap from the most recently created snapshot, if this has not already been done. An in-core bit map records the completed updates to avoid repeating them unnecessarily. This ensures that the combination of the active bitmap and the summary file will consistently identify all blocks that are currently in use. Additionally, the summary file is updated to reflect the effect of any recent snapshot deletions when freeing a block in the active file system. This allows reuse

of blocks that are now entirely free. After updating the summary file following a snapshot creation or deletion, the corresponding bit in the fsinfo block is adjusted. In a seventh aspect of the invention, the algorithm for deleting a snapshot involves examining the snapmaps of the deleted snapshot and the snapmaps of the next oldest and next youngest snapshot. A block that was used by the deleted snapshot but is not used by its neighbors can be marked free in the summary file, as no remaining snapshot is using it. However, these freed blocks cannot be reused immediately, as the snapmap of the deleted snapshot must be preserved until summary updating is complete. During a snapdelete, free blocks are found by using the logical OR of the active bitmap, the summary file, and the snapmaps of all snapshots for which post-deletion updating is in progress. In other words, the snapmap of the deleted snapshot protects the snapshot from reuse until it is no longer needed for updating. In the preferred embodiment, the invention is operative on WAFL file system. However, it is still possible for the invention to be applied to any computer data storage system such as a database system or a store and forward system such as cache or RAM if the data is kept for a limited period of time.